VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2009 Vít Klimenko

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

CMS redakční systém

CMS content management system, compare, development and implementation

2009 Vít Klimenko

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2009 .....

Na tomto místě bych rád poděkoval svému vedoucímu bakalářské práce panu Ing. Radoslavu Fasugovi, za řadu podmětných nápadů a rad při návrhu a realizaci systému.

Abstrakt

Tato práce se zabývá analýzou a implementací jednoduchého CMS redakčního systému s využitím moderních internetových technologií. Sestavuje přehledovou studii volně dostupných redakčních systémů, zabývá se jejich vlastnostmi a možnostmi rozšíření. Cílem vlastní implementace je vytvoření univerzálního jádra webové aplikace, které bude využito ve vlastním redakčním systému a snadno implementovatelné pro další webové aplikace. Celá implementace bude realizována na aplikačním rámci Zend Framework, který zajišťuje trendy moderního programování jako je dodržení vícevrstvé architektury systému, objektového programování, zajištění multiplatformního databázového přístupu a mnoho dalších užitečných vlastností.

Klíčová slova: informační systém, redakční systém, systém pro správu obsahu, elektronické publikování, webové aplikace, internetové technologie, optimalizace pro vyhledávače, zend framework, multiplatformní databázový přístup

Abstract

The purpose of this theses is analysis and implementation of a simple CMS content management system using modern Internet Technologies. Writes overview study about OpenSource available editorial systems, deals with their properties and the possibility of extensions. The importance of self-implementation is the establishment of a universal core web application to be used in the editorial system. All implementation will be realized on the application of the Zend framework, which provides a modern programming trends such as compliance with the multi-layer architecture, object-oriented programming, multiplatform database access and many other useful features.

Keywords: information system, editorial system, content management system, electronic publishing, web applications, internet technologies, search engline optimalization, zend framework, multiplatform database access

Seznam použitých zkratek a symbolů

Ajax	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
CSS	–	Cascading Style Sheet
CMS	–	Content Management System
GUI	–	Graphical User Interface
HTTP	–	HyperText Transfer Protocol
IS	–	Informační systém
JS	–	JavaScript
PHP	–	PHP: Hypertext Preprocessor (původně Personal Home Page)
RS	–	Redakční systém
RSS	–	Rich Site Summary (v0.91), Really Simple Syndication (v2.0)
SEO	–	Search Engline Optimalization
SŘBD	–	Systém řízení báze dat
SQL	–	Structured Query Language
UML	–	Unified Modeling Language
URL	–	Unified Resource Locator
WWW	–	World Wide Web
WYSIWYG	–	What You See Is (all) What You Get
XHTML	–	eXtensible HyperText Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V úvodních kapitolách tato práce obsahuje samotnou problematikou redakčních systémů a systémů pro správu obsahu. Uvádí jejich vlastnosti a požadavky na funkcionalitu. Dále popisuje přehledovou studii volně dostupných redakčních systémů, jaké jsou jejich výhody a nevýhody, možnosti rozšíření a příklady využití. Popisuje vlastnosti webových aplikací jako jsou moderní trendy vývoje internetových aplikací a základní funkce redakčních systémů.

Ze shromážděných informací bude vytvořena analýza a vlastní řešení CMS redakčního systému. K vývoji bude využit programový aplikační rámec Zend Framework, který obsahuje komponenty jako je zajištění vícevrstvé architektury systému, multiplatformní databázový přístup, tvorbu uživatelského prostředí a především flexibilita prezentační části jako je snadná možnost změny vzhledu a jednoduché programování modulů.

2 Informační redakční systémy

Mezi dnešní moderní webové technologie patří dynamicky generované stránky. Práce obsahuje stručný náhled do současných trendů vývoje webových aplikací. Jedním z hlavních cílů této práce je seznámení se s problematikou CMS redakčních systémů a shrnutím stávajících řešení.

2.1 Moderní webové technologie

Od doby, kdy internetové služby poskytovaly informace v podobě statických webů již uplynula řádka let. Dnes webové technologie nabízejí velmi rozsáhlé možnosti tvorby aplikací, které zajišťují přívětivá uživatelská prostředí a funkcionality připomínající desktopové aplikace.

Největší změna v oblasti webu nastala s příchodem skriptovacích jazyků běžících na straně webových serverů a jejich propojení s daty uložnými v databázových úložištích a souborech. Díky těmto možnostem začaly vznikat rozsáhlé informační systémy a webové aplikace jako jsou elektronické obchody, vyhledávací systémy, redakční systémy a mnoho dalších aplikací, se kterými se denně shledáváme v prostředí internetu.

Pro skriptování na straně klienta resp. v klientských webových prohlížečích vzniklo několik technologií, které zajišťují tvorbu přívětivých uživatelských rozhraní s možností formátování obsahů pomocí (X)HTML v kombinaci s CSS[1], dále programování pro elementy stránky napodobující desktopové aplikace (skriptovací jazyk JavaScript) a také v posledních letech moderní trend AJAX[2], což je vzájemné propojení několika webových technologií (XHTML, CSS, XML a JS propojující tyto technologie) pro vytváření interaktivních webových aplikací. Tyto možnosti poskytují tvorbu aplikací, které například simultálně komunikují se SŘBD, dovolují okamžitě reagovat na vstupy uživatele aniž by musela být stránka průběžně obnovena (reload). Těchto vlastností je využito ve vlastním systému pro filtrování uživatelských vstupů (zjištění unikátnosti přihlašovacího jména).

Další usnadnění vývoje webových aplikací nabízejí tzv. programové aplikační rámce neboli frameworky. Vlastní redakční systém je realizován na aplikačním rámci Zend[3], jehož výhody a vlastnosti budou popsány v dalších kapitolách. Aplikační programové rámce usnadňují a urychlují vývoj softwarových produktů.

Aplikační rámec umožňuje se soustředit na vývoj funcionality aplikace, místo toho aby vývojáři řešili jakým způsobem bude aplikace implementována. Vývojáři se nemusí zaměřovat na programování základních funkcionalit systému jako je například zpracování požadavků z URL adres, práce s (X)HTML komponenty, ověřování uživatelských vstupů, práce se SŘBD a mnoho dalších funkcionalit, ale zaměří se na programování požadovaných systémových vlastností. Programátor využívá již předprogramované moduly, rozhraní a komponenty. Toto řešení zajišťuje větší bezpečnost a eliminuje množství

[1] Cascading Style Sheets – jazyk kaskádových stylů popisuje způsob zobrazení webové stránky napsané v jazycích (X)HTML či XML

[2] Asynchronous JavaScript and XML – je to technologie pro vývoj interaktivních webových aplikací, jehož využití značně posunul vývoj internetových technologií

[3] `http://framework.zend.com/` – oficiální webová stránka projektu Zend Framework

chyb aplikace, které se během vývoje vyskytují a musí se ve fázi testování odstranit. Další výhodou je využití komponent, jejíž vývoj by byl velmi zdlouhavý, komplikovaný či finančně náročný. Příkladem takových komponent mohou být rozhraní *Google API* pro komunikaci s aplikacemi Google jako jsou Google Maps, Google Picasa a další.

2.2 Informační systém

Informační systém lze chápat jako systém vzájemně propojených informací a procesů, které mezi těmito informacemi komunikují. Procesy znamenají funkce, které zajišťují zpracovávání vstupních informací do systému a transformují je na informace ze systému vystupující. To znamená zajištění funkcionality pro získávání, přenos, uložení, zpracování a výslednou distribuci informací. Samotný IS ovlivňuje tzv. „okolí“. Okolí informačního systému tvoří objekty, které při změně svých vlastností ovlivní stav samotného systému a mezi ně také patří objekty, které změní své vlastnosti v závislosti na systému.

Informační systém tedy znamená softwarové vybavení firmy, instituce či organizace, které může zajišťovat samotné řízení podniku nebo poskytuje informace zaměstnancům, řídícím pracovníkům, zákazníkům a dalším uživatelům. Díky těmto vlastnostem IS zajišťují vykonávání určité aktivity a rychlou dostupnost informací. Mezi tyto procesy patří například plánování, publikace informací, zjištění stavu zakázek a mnoho dalších úkonů.

2.3 Webové informačních systémy

Webový informační systém je specifický typ informačního systému, určeného pro provoz v internetové nebo intranetové síti. Nejedná se ovšem o jednoduché internetové stránky. Webový informační systém bývá ve většině případů velmi rozsáhlá komplikovaná aplikace. Tyto systémy se vyvíjejí tak, aby v maximální míře vyhovovali uživatelům a zákazníkům. Systémy řeší problematiku autentizace, autorizace, modularizace, hierarchického řazení uživatelů a spoustu dalších činností.

2.3.1 Odstranění nestavovosti

Internetový protokol HTTP je bezstavový[4] protokol zajišťující výměnu hypertextových dokumentů v oblasti internetu a intranetu. Při dokončení přenosu souboru se spojení mezi serverem a klientem ukončí a tím se ztratí aktuální stav, ve kterém se webový informační systém právě nacházel. Toto je základní rozdíl v programování aplikací oproti prostředí klient/server (síťových aplikací). Jednotlivé HTTP požadavky a jejich prováděné operace nemají mezi sebou žádnou návaznost. Jedná se tedy o izolované přístupy k samotnému webovému informačnímu systému. Jako příklad se může uvézt editace dat v datovém úložišti, nejprve se musí dotázat na výběr dat, samotná editace, kontrola a poté zpětné uložení. Je nutné zajistit přesun dat mezi těmito kroky. Webové technologie

[4]tzv. „stateless“ bezstavový protokol, tzn. že v protokolu není nijak zabudované, jak uchovávat stav mezi dvěma transakcemi. Když uživatel požaduje nějakou stránku a poté jinou, HTTP protokol neposkytuje žádnou možnost, jak by se mohlo zjistit, že oba požadavky přišly od stejného uživatele.

nabízí dva způsoby tohoto řešení. Prvním řešením je přenos parametrů mezi jednotlivými požadavky, které zajišťují HTTP metody *GET*[5] a *POST*[6].

Dalším typem řešení je sklad parametrů relací neboli sezení. Ty se rozdělují na dva typy, prvním typem jsou tzv. *COOKIES*. Cookies jsou malé textové soubory uložené na straně klienta a umožňují webovým serverům ukládat a načítat data z klientova pevného disku. Druhým typem sezení jsou soubory uložené na straně serveru neboli *SESSIONS*. Ty umožňují identifikaci klienta a jeho pohyb na serveru. Sessions funguje na principu vygenerování unikátního uživatelského čísla ID[7], které je uživateli přiřazeno. Díky tomuto číslu je systém schopen identifikovat konkrétního klienta. Výhodou oproti cookies je neomezené předávání počtu proměnných. Klientský prohlížeč může mít počet uložených cookies omezenen nebo dokonce úplně zakázán, proto je lepší a bezpečnější variantou používat ukládání relací pomocí *SESSIONS*. Dalším důvodem jsou bezpečnostní rizika. U Cookies se posílají všechna data při každém požadavku, zatímco Session posílá jen potřebné datové informace.

Díky těmto technikám uložení dat se potlačí bezstavovost HTTP protokolu. Praktické využití může být např. pro systémové uložení právě přihlášených uživatelů či ukládání obsahu nákupního košíku v elektronických obchodech po takovou dobu, dokud zákazník nepotvrdí konečnou objednávku. V implementaci systému je využíváno ukládání identity přihlášených uživatelů pomocí *SESSIONS* a pro jednoduchou obsluhu těchto proměnných je použita knihovna *Zend_Session* aplikačního rámce *Zend Framework*.

2.3.2 Autorizace a autentizace

Autentizace a autorizace jsou jedním z nejčastěji řešených problémů ve webových informačních systémech.

Autentizace znamená prokázání totožnosti přihlášeného uživatele. Ve většině případů se autentizace provádí pomocí ověření unikátního uživatelského jména a hesla. U složitějších systémů, ve kterých je důležitá vysoká míra bezpečnosti jako je internetové bankovnictví se provádí ověření identity i pomocí dalších ověřovacích technik. Může být využito ověřování identity pomocí přiděleného certifikátu potvrzeného certifikační autoritou nebo unikátního klíče zaslaného v SMS[8] zprávě na mobilní telefon klienta. Další typy autentizací nabízí např. webový server Apache s integrovaným modulem HTTP autentizace. Toto řešení ovšem není vhodné pro rozsáhlé webové informační systémy. Ve vlastním řešení je využito autentizace pomocí komponenty *Zend_Auth*.

Autorizace je proces, u kterého se uživatelům přidělují oprávění pro provádění konkrétních akcí. Tento proces zajišťuje systém oprávnění, který definuje kategorie uživatelů provádějících třídy určitých úloh a tím jsou rozlišováni různě privilegovaní uživatelé.

[5]Metoda GET je nejjednodušší metodou pro zasílání dat přes HTTP protokol. Prakticky pokaždé, když se načítá stránka ze serveru a nebyly odeslány formulářová data pomocí metody POST, používá se k obdržení stránky ze serveru právě tato metoda.

[6]Metoda POST funguje v podstatě stejně jako metoda GET, ale u POST máme možnost za hlavičkami poslat skriptu data. Touto metodou se např. odesílají data z formulářů.

[7]Session ID neboli Session Token

[8]Short Message Service – GSM služba krátkých textových zpráv

Tento systém přidělování uživatelských práv je řešen pomocí ACL[9]. Ukázka těchto ACL listů je v implementaci systému a zdrojových kódech s komentáři.

2.3.3 Vícejazyčnost systému a nastavení národního prostředí

Vícejazyčnost je užitečná vlastnost pro systémy, které využívají uživatelé z různých zemí. Systém by měl podporovat uživatelskou volbu jazykového prostředí alespoň pro základní světové jazyky jako je angličtina, němčina, francouzština a další. Systém by měl mít navrženu vícejazyčnost takovým způsobem, aby se do něj dal kdykoliv přidat další jazyk a to bez zásahu do struktury systému.

Dalším důležitým aspektem je nastavení národního prostředí. Pouze přeložení výstupů systému do konkréního jazyka nestačí. V jednotlivých jazykových regionech jsou různé konvence pro jména, příjmení, formátování čísel, data, času a měn. Toto je další nutná vlastnost systému, která musí být vyřešena.

Jako příklad se může uvést formát data 2009-05-07, které je uloženo v datovém úložišti. Pro české jazykové prostředí musí systém zobrazovat datum ve formátu, které je dané českými normami a tím je např. 7. května 2009 či 7. 5. 2009. Zatímco americký jazykový formát je May 5, 2009 nebo 5/7/09 a zapsání britského formátu je odlišné v pořadí měsíce a způsobu zobrazení čísla 7th May 2009. K těmto konverzím se opět využívá další knihovny aplikačního rámce.

2.3.4 Ovládání a uživatelské prostředí

Komponent jazyka (X)HTML sice není tolik, na jaké jsou programátoři zvyklí při programování desktopových aplikací, ale poskytují mnoho způsobů rozšíření. Využívá se základních komponent jazyka (X)HTML a jejich případná funkcionalita se může rozšířit pomocí JavaScriptu. Uživatelské prostředí lze snadno přizpůsobit požadavkům zákazníka, ať je to volba struktury uživatelského prostředí, grafického návrhu, či samotné funkcionality. Pro informační systémy se vytváří spíše jednoduché uživatelské prostředí, neklade se důraz na grafické prvky, jako je tomu u reklamních webových stránek a prezentací. Hlavní důraz je kladen na funkčnost a jednoduché prostředí, ve kterém se uživatel snadno orientuje a zajišťuje jednoduché ovládání a rychlou dostupnost požadovaných informací.

2.4 Systémy pro správu obsahu

Systémy pro správu obsahu (dále jen CMS) jsou aplikace určené k vytváření, úpravě, hledání a publikaci různých typů digitálních médií a elektronických textů. Jejich největší využití je především v prostředí internetu. Lze publikovat videa, hudební materiály, fotografie, novinové články a mnoho dalších typů digitálních médií. Nejdůležitějšími vlastnostmi těchto systému je zajištění jednoduché obsluhy uživatelů (redaktorů, autorů). Vkládání a editace médií se provádí většinou pomocí vizuálních editorů (WYSIWYG)

[9] Access Control List – zajišťují řízení přístupu k požadovaným akcím. Definují seznam objektů, osob či skupin a těm jsou přiřazována oprávnění neboli „role“ akcí.

a nevyžadují po uživatelích žádnou znalost programování a značkovacích formátovacích jazyků jako jsou (X)HTML a CSS.

CMS mohou být jednoduché aplikace obsluhované jediným uživatelem a jejich využití je vhodné pro občasnou editaci obsahu na internetových stánkách (malé firmy, osobní webové prezentace, blogy[10]). Mohou to být také rozsáhlé víceuživatelské systémy obsluhující uživatelé, kteří mají přidělená práva a role pro obsluhu funkcionalit systému. CMS nemusí být určeny jen pro publikaci informací a digitálních médií, mohou také obsahovat spoustu rozšiřitelných modulů jako jsou katalogy, ankety, fotogalerie, diskuzní fóra, komentáře a další rozšíření.

2.5 Redakční systémy

Redakční systém (dále jen RS) je specifický typ systému pro správu obsahu. Jeho využití je především v prostředí internetu na redakčních serverech. Již slovo „redakční" předjímá uplatnění těchto informačních systémů. Používají se k publikování elektronických textů, novinových a odborných článků. Příkladem redakčních serverů jsou `http://root.cz`, `http://idnes.cz` a spousta dalších. Servery, na kterých běží redakční systémy denně navštěvují tisíce čtenářů, kteří využívají rychlou dostupnost aktuálních informací a zpráv. Systémy zajišťují víceuživatelský přístup a přidělování práv uživatelům (čtenáři, autoři, správci atd.). Jako příklad se mohou uvést redaktoři, kteří vytvářejí a editují články, dále to mohou být korektoři provádějící schvalování a opravy chyb, internetový čtenáři a také správci, zajišťující administraci samotného systému.

2.6 Přehledová studie dostupných CMS, redakčních a blogovacích systémů

Vývojářské firmy a samotní vývojáři vytvářejí opensource i komerční řešení. Z funkčního hlediska mají systémy různé vlastnosti a cíl využití. Některé systémy slouží jako veřejné internetové encyklopedie, některé jsou určeny pro jednoduché webové prezentace. Jsou dostupná rozsáhlá řešení, které vyvíjí několika-členné vývojářské týmy. V následující kapitole bude popsáno několik hotových řešení a jejich výhody a nedostatky, možnosti využití, či případná technologická omezení.

2.6.1 Systémy Wiki

Wiki jsou speciální typy webových aplikací, které slouží jako multiuživatelské redakční systémy. Wiki je v podstatě aplikace pro vytváření, prohlížení a prohledávání navzájem propojených informací. Jejich hlavní využití je například ve světově známé internetové encyklopedii Wikipedia. Wikipedia je mnohojazyčná encyklopedie, na jejíž tvorbě obsahu se může podílet každý, kdo chce tuto encyklopedii rozvíjet a rozšiřovat. Kdokoli

[10] Web Log zkráceně blog – je webová aplikace sloužící jako webový zápisník. Využívá se jako internetový zápisník či deník, nebo jako jednoduchý systém, ve kterém autoři publikují své myšlenky, výsledky práce, návody a spoustu dalších užitečných informací.

s přístupem na internet může vytvořit v encyklopedii vlastní článek nebo stávající články upravit, rozšířit a v případě nekorektního obsahu jej může celý přepsat.

Pro editaci obsahu se používá zjednodušený značkovací jazyk „wikitext“. Z tohoto značkovacího kódu je vygenerován výsledný (X)HTML kód zobrazující se v klientském prohlížeči. Důvodem vytvoření tohoto značkovacího jazyka byla komplikovanost (X)HTML kódu. Umožňuje autorovi obsahu se více zaměřit na samotný obsah článku než na psaní strukturovaného (X)HTML dokumentu. Wiki systémy dovolují psát články z různých oborů, zajišťují vkládání obrázků, matematických vzorců a spousty dalších možností nasazení.

Jedním z nejrozšířenějších Wiki systémů je aplikace MediaWiki[11]. Výhodou systému MediaWiki je možnost nasazení na většinu webových serverů podporujících skriptovací jazyk PHP a databázové systémy MySQL, PostgreSQL či SQLite. Tím zajišťuje širokou možnost nasazení, protože výše zmíněné technologie a aplikace jsou v prostředí internetu nejrozšířenější.

Výhody:

- ideální pro vytvoření manuálových stránek a encyklopedie
- použití rozšířeného skriptovacího jazyka PHP
- možnost využití několika SŘBD systémů (MySQL, PostgreSQL a SQLite)

Nevýhody:

- k editaci textů se využívá značkovacího jazyka „wikitext“, který se musí uživatelé naučit (existuje ovšem možnost rozšíření o WYSIWYG editor)

2.6.2 Joomla!

„Joomla!“ je další z volně šířitelných CMS systémů. V základním balíku se jedná o velmi rozsáhlou aplikaci, na jejímž vývoji se podílí široká komunita vývojářů. Systém je snadno rozšířitelný o moduly neboli „pluginy“[12]. Mezi základní funkce systému patří správa kategorií pro zařazování článků, správa uživatelů, psaní novinek, blogy, ankety, možnost zobrazení stránek určených pro tisk, vyhledávání v článcích, multijazyčnost a také možnost odběru článků pomocí RSS kanálů. Jednou z nevýhod „Joomla!“ je její omezení v multiplatformnosti pro běh databázových systémů. „Joomla!“ podporuje pro svůj chod jen databázový systém MySQL, ostatní systémy SŘBD nejsou podporovány jak tomu bylo ve výše zmíněném Wiki systému. Ve vlastním řešení je zajištění běhu pro multiplatformnost databázových systémů jednou z hlavních podmínek vývoje.

[11] MediaWiki – opensource software napsaný v jazyce PHP. Světová encyklopedie Wikipedia tento Wiki systém využívá. MediaWiki je ke stažení na internetové adrese `http://www.mediawiki.org`.

[12] Pro systém „Joomla!“ existuje více než 4000 modulárních rozšíření, jedná se tedy o velmi rozsáhlé softwarové řešení vyvíjející širokou komunitou vývojářů.

Výhody:

- rozsáhlá možnost využití
- široká komunita uživatelů a vývojářů
- velké množství rozšiřujících pluginů
- již v základní verzi je systém velmi rozsáhlý (což může být občas omezující)

Nevýhody:

- podpora pouze jednoho databázového systému (MySQL)
- systém je napsán v jazyce PHP4, který je již zastaralým (aktuální verze je PHP5)

2.6.3 WordPress

WordPress je specifickým typem CMS systému, který funguje jako internetový zápisník neboli „blog“. Slouží pro vkládání zápisků, jednoduchých článků a nápadů konkrétního uživatele, který tento systém využívá a chce své zápisky zpřístupnit veřejnosti. Systém je navržen tak, aby si uživatel mohl snadno pozměnit vzhled blogu, editovat jakýkoli obsah pomocí vizuálního editoru WYSIWYG. Jednotlivé zápisky jsou zařazovány do kategorií a také podle data vydání do tzv. „archivů“. Výhodou řešení je rozsáhlá modulární rozšiřitelnost pomocí pluginů. Mezi užitečná rozšíření patří například možnost napojení na služby *Google API* nebo rozšíření o fotogalerie apod. Nevýhodou tohoto řešení je opětovná závislost na databázovém systému MySQL a tím není zaručena přenositelnost mezi databázovými platformami. Starší verze systému WordPress byla také portována na databázový systém PostgreSQL, ale tento projekt byl po čase ukončen.

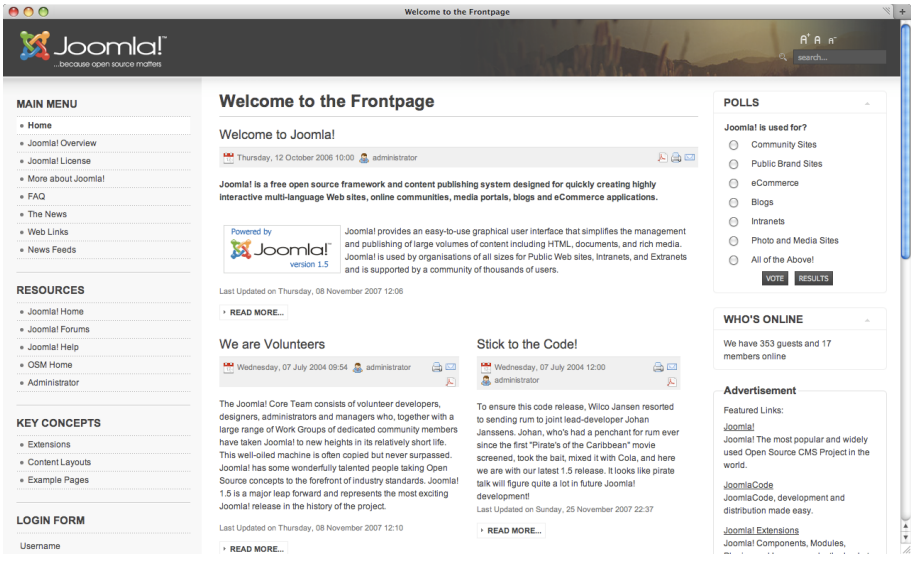
Výhody:

- množství dostupných modulů rozšíření
- široká základna uživatelů a vývojářů
- k WordPress je vydáno několik knižních publikací

Nevýhody:

- systém je napsán v PHP4 a podporuje jen databázový systém MySQL
- portovaná verze pro podporu PostgreSQL se již nevyvíjí
- využití aplikace jen pro internetový zápisník

Obrázek 1: Systém Wiki - encyklopedie Wikipedia



Obrázek 2: Systém „Joomla!“

2.6.4 Drupal

Systém Drupal je dalším CMS systémem, který se řadí mezi nejobsáhlejší. Využití tohoto systému je v oblasti malých osobních prezentací i pro velké korporátní weby. Může sloužit jako redakční systém, blog či internetové fórum. Systém je navržen pro možnost modulárních rozšíření. Drupal je označován jako „Content Management Framework" neboli aplikační rámec pro správu obsahu, a to díky své rozšířitelnosti. Drupal v základní verzi obsahuje výše zmíněné požadavky CMS systémů jako je správa obsahu, kategorií, uživatelských účtů a spoustu dalších funkcí. Výhodou tohoto systému oproti výše zmíněným je rozsáhlejší podpora databázových systémů. Při instalaci Drupalu je na výběr ze dvou databázových systémů a to z *MySQL* a *PostgreSQL*. Drupal využívá databázová úložiště pro ukládání obsahu a nastavení. Systém vyžaduje pro svůj běh podporu skriptovacího jazyka *PHP* (od verze 4.3.5 a vyšší). To sice zajišťuje větší možnost nasazení na webových serverech, ale dnes je již *PHP* verze 4 zastaralá. Přechází se na verzi *PHP5*, která je sice s předešlou verzí kompatibilní, ale má oproti původní verzi vestavěnou kompletní podporu objektově orientovaného programování (OOP)[13].

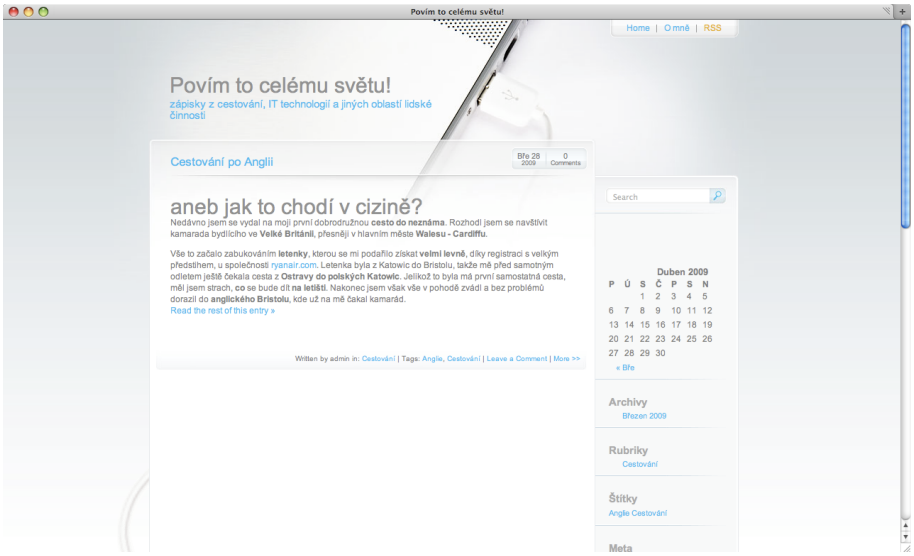
Výhody:

- rozsáhlá možnost využití
- široký výběr rozšiřujících modulů
- volba ze dvou databázových systémů (MySQL a PostgreSQL)
- je součástí repozitářů několika Linuxových distribucí
- široká uživatelská a vývojářská základna
- jednoduchá instalace

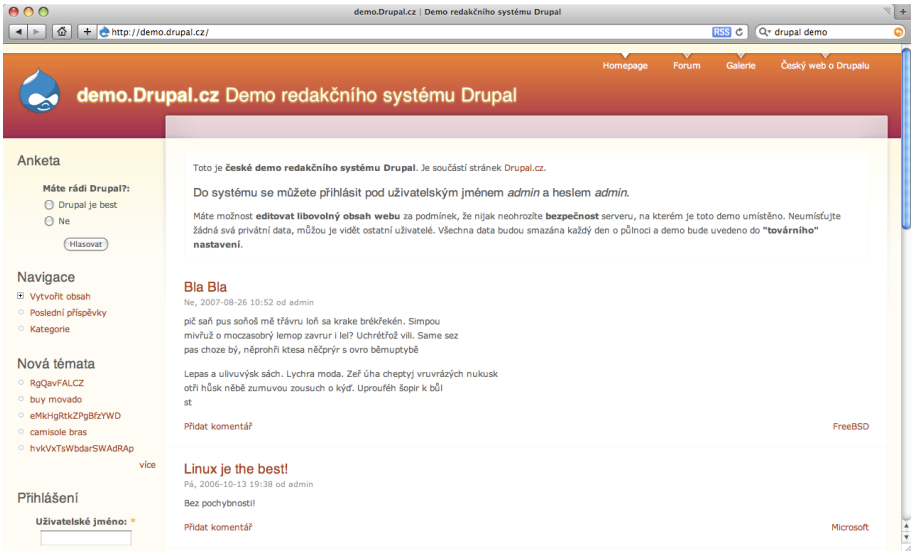
Nevýhody:

- veliká robustnost systému
- systém využívá skriptovací jazyk PHP4 (PHP5 je s verzí PHP4 kompatibilní)

[13]Object-oriented programming neboli objektově orientované programování – je způsob programování, ve kterém se řešení programových úloh zakládá na objektech reálného světa. Vytváří se v programu třídy, ze kterých se za běhu programu vytvářejí objekty a ty napodobují chováním a vlastnostmi reálné objekty. Mezi programovací jazyky podporující OOP patří C++, Java, PHP5, C#, Python a další.



Obrázek 3: Blogovací systém WordPress



Obrázek 4: Systém Drupal

2.7 Přehled základních funkcí CMS redakčních systémů

Základní funkce RS mohou být z hlediska požadavků rozdílné. Tyto základní funkce se využívají nejčastěji:

Editace článků (obsahu) – nejdůležitější část redakčního systému. Autor článku se přihlásí do systému, zvolí kategorii, do které bude článek zařazen, a poté vkládá obsah článku. U složitějších systémů mají možnost editace článků jen uživatelé s oprávněním editace, jako jsou redaktoři či korektoři. Systém musí podporovat snadnou správu informací bez znalosti tvorby webových stránek, což je jednou z nejdůležitějších vlastností RS. K samotné editaci se využívá vizuální WYSIWYG editor.

Správa kategorií – RS podporují zařazování článků do kategorií. Příkladem jsou novinové redakční servery, na kterých jsou zpravodajské články řazeny do kategorií podle typů článků (např. domácí, zahraniční, technické).

Systém uživatelských práv – zavádění rolí uživatelů pomocí přidělených práv (například správce systému, redaktoři – autoři článků, registrovaní uživatelé, čtenáři).

Vizuální editor – pro formátování a editaci článků se nejčastěji využívá grafický vizuální WYSIWYG editor. Tento editor zajišťuje jednoduché formátování textů a vkládání obrázků. Usnadňuje uživatelům jednoduchou editaci a především zamezuje vkládání chyb při formátování textů pomocí značkovacích jazyků jako je (X)HTML. WYSIWYG editor se ovládá podobně jako známé kancelářské textové editory (Microsoft Word, OpenOffice.org Writer).

Vyhledávání – prohledávání článků pomocí fulltextového vyhledávání. Jde o nejefektivnější typ vyhledávání podle zadaných kritérií, klíčových slov. K fulltextovému vyhledávání se využívá databázových systémů a textových souborů, ve kterých jsou informace uloženy. V databázových systémech se pro zrychlení doby vyhledávání indexují záznamy a tím je zajištěno zrychlení vyhledané informace, aniž by musely být soubory kompletně procházeny.

Multijazyčnost – mnoho redakčních systémů publikuje články v několika jazykových formách. Systém musí podporovat vkládání vícejazyčných článků a podporu národního prostředí.

Odběr článků – uživatelé internetu často využívají možnosti odběru článků pomocí RSS kanálů. Pokud tento odběr webový prohlížeč či čtečka podporují, čtenář si může zvolený RSS kanál své oblíbené stránky zaregistrovat. Díky této technologii nemusí čtenáři denně navštěvovat své oblíbené stránky a hledat nově vložené články. O toto se postará jejich RSS čtečka, která pravidelně prochází zaregistrované RSS kanály a ihned při vložení nového článku bude čtenáře informovat. Článek se čtenáři objeví v čtečce jako nově doručený email či zpráva. Čtenář si může článek ihned přečíst, aniž by musel zdlouhavě procházet stránky. Ve vlastním řešení bude integrována publikace nově vložených obsahů pomocí RSS.

Optimalizace pro vyhledávače – moderní trendy internetových aplikací vyžadují tzv. „SEO optimalizaci" neboli optimalizaci pro vyhledávače. Jedná se o vytváření webových stránek a jejich obsahů takovým způsobem, aby byly vhodné pro automatické zpracování a zařazování do internetových vyhledávacích systémů. Hlavním cílem optimalizace je získání ideální pozice ve vyhledávačích a především vysoká pravděpodobnost vyhledání požadované informace po zadání klíčových slov.

3 Analýza řešení

Analýza je jedním z nejdůležitějších kroků při tvorbě softwaru. Popisuje datové struktury a proces vývoje softwarového díla. Pro vytvoření kvalitní datové vrstvy je důležité podrobné zpracování datové analýzy CMS redakčního systému. První fází datové analýzy je vytvoření konceptuálního datového modelu, ve kterém se popisuje návrh struktury datových entit, atributů, vztahů mezi entitami a vytvoření podrobného datového slovníku obsahujícího datové typy a integritní omezení samotných atributů. Další fází analýzy je vytvoření několika vývojových diagramů v jazyce UML popisujících případy užití systému ze strany aktérů, třídní diagram relačního datového modelu a sekvenční diagramy zobrazující stavy při běhu systému.

3.1 Konceptuální datový model

Zajišťuje výsledné konceptuální schéma struktury databáze.

3.1.1 Lineární zápis typů entit

Značení:

- **primární atribut** – podtržený zvýrazněný text
- cizí klíč – podtržený text
- Y – ano
- N – ne
- IO – integritní omezení
- YYYY – rok, MM – měsíce, DD – dny
- HH – hodiny, mm – minuty, ss – sekundy
- NN – číselné hodnoty

users (**idu**, name, surname, username, password, email, role, created, updated, active, comment)

– klíčové atributy: idu
– cizí klíče: –

articles_category_tree (**idc**, top, order, tree)

– klíčové atributy: idc
– cizí klíče: –

articles_category_name (**idn**, idc, idl, idu, name)

– klíčové atributy: idn
– cizí klíče: idc, idl, idu

articles (**ida**, idc, idl, idu, title, text, keywords, comment)
- – klíčové atributy: ida
- – cizí klíče: idc, idl, idu

languages (**idl**, lang)
- – klíčové atributy: idl
- – cizí klíče: –

users_logging (**idg**, idu, date_login, date_logout, ip)
- – klíčové atributy: idg
- – cizí klíče: idu

3.1.2 Popis jednotlivých entit

users – tabulka uživatelů, obsahuje informace o uživatelích, přihlašovací jména, hesla, uživatelské role a spoustu dalších informací.

articles_category_tree – obsahuje informace o zařazení a pořadí kategorií jednotlivých článků. Položka *tree* obsahuje textový řetězec vytvořený z hodnot *top*, *order* a nadřazených kategorií. Tento způsob stromové struktury zjednodušuje SQL dotazy, které se využívají k získávání dat z databázového úložiště. Nemusí se vytvářet složené SQL dotazy.

articles_category_name – názvy jednotlivých kategorií, tabulka je vytvořena k uložení názvu jedné kategorie v několika jazykových formách.

articles – obsah článku zařazeného v jednotlivých kategoriích. Obsah je v tabulce uložen v několika jazykových formách, pokud je v systému zvoleno více jazykových prostředí.

languages – uložení jazyků systémových jazykových prostředí.

users_logging – ukládání systémových informací o přihlašování a odhlašování uživatelů (čas přihlášení a odhlášení s IP adresou).

3.1.3 Lineární zápis vztahů mezi entitami

Popisuje vztahy mezi datovými entitami systému. V tabulce jsou vyjádřeny názvy vztahů, seznam vztahů a vzájemných typů mezi entitami.

Název	Entity	Typ	Popis
LOGS	users, users_logging	1 : N	logování uživatelů
CREATE_ARTICLE_CATEGORY	users, articles_category_name	1 : N	uživatel editující kategorii
CREATE_ARTICLE	users, articles	1 : N	uživatel editující článek
CREATE_LANG	users, languages	1 : N	uživatel editující jazyk
CATEGORY_INSERT	articles_category_tree, . . . _name	1 : N	zařazení kategorie
CATEGORY_LANG_NAME	languages, articles_category_name	1 : N	jazykové mutace
ARTICLE_CLASSFICATION	articles_category_tree, articles	1 : N	zařazení článku
ARTICLE_LANG	articles, languages	1 : N	jazyková mutace

3.2 Datový slovník

Datový slovník je seznamem datových objektů v databázi, znázorňuje jména a datové typy atributů, vztahů a integritních omezení.

Viz příloha A.

3.3 Vizuální zobrazení návrhu aplikace

Pro vizuální zobrazení návrhu software se využívá modelovacích technik UML diagramů. Využitím modelovacího jazyka UML se znázorní návrh, specifikace a dokumentování systému.

3.3.1 Kontextový diagram uživatelských rolí

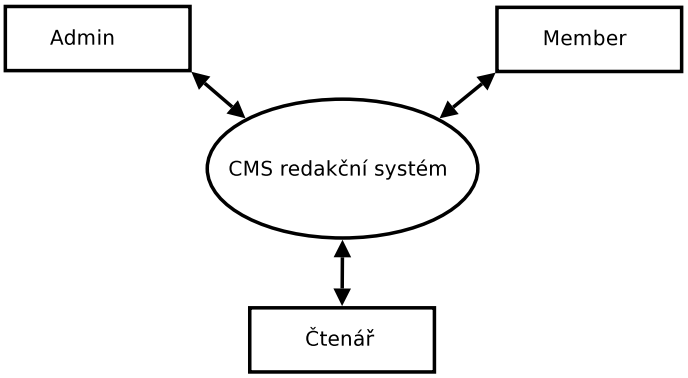
Kontextový diagram uživatelských rolí znázorňuje přístup uživatelů k systému. V aplikaci jsou zavedeny tři typy uživatelů, prvním typem jsou neregistrovaní uživatelé, kteří mají oprávnění procházet kategorie, články a odebírat RSS. Dalším typem jsou registrovaní uživatelé „member“, kteří mají oprávnění k vytváření kategorií a článků (autoři). Posledním typem uživatelů jsou „admin“ neboli administrátoři (správci systému), kteří mají nejvyšší oprávnění. Administrátoři mohou vytvářet, editovat kategorie a články, mají na starost správu systému a vytváření uživatelských účtů.

Značení:

čtenář – nepřihlášený uživatel, práva k procházení kategorií a článků, odběr pomocí RSS.

member – vytváření, editace kategorií a článků (member je registrovaný člen).

admin – má nejvyšší oprávnění – editace, vytváření kategorií a článků, správa uživatelských účtů, správa jazykových prostředí a systému.



Obrázek 5: Kontextový diagram

3.3.2 Diagram případů užití

Diagram případů užití (Use Case Diagram – viz obr. 6) slouží ke grafickému znázornění systémových funkcí a oprávnění aktérů (přístupu uživatelů k těmto funkcím).

3.3.3 Třídní diagram relačního datového modelu

Třídní diagram (Class Diagram) relačního datového modelu slouží k zobrazení datových entit (viz obr. 7). Tento diagram je zvolem z důvodu podrobnějšího zápisu nežli známé ER diagramy (Entity-relationship diagram).

3.3.4 Stavové diagramy

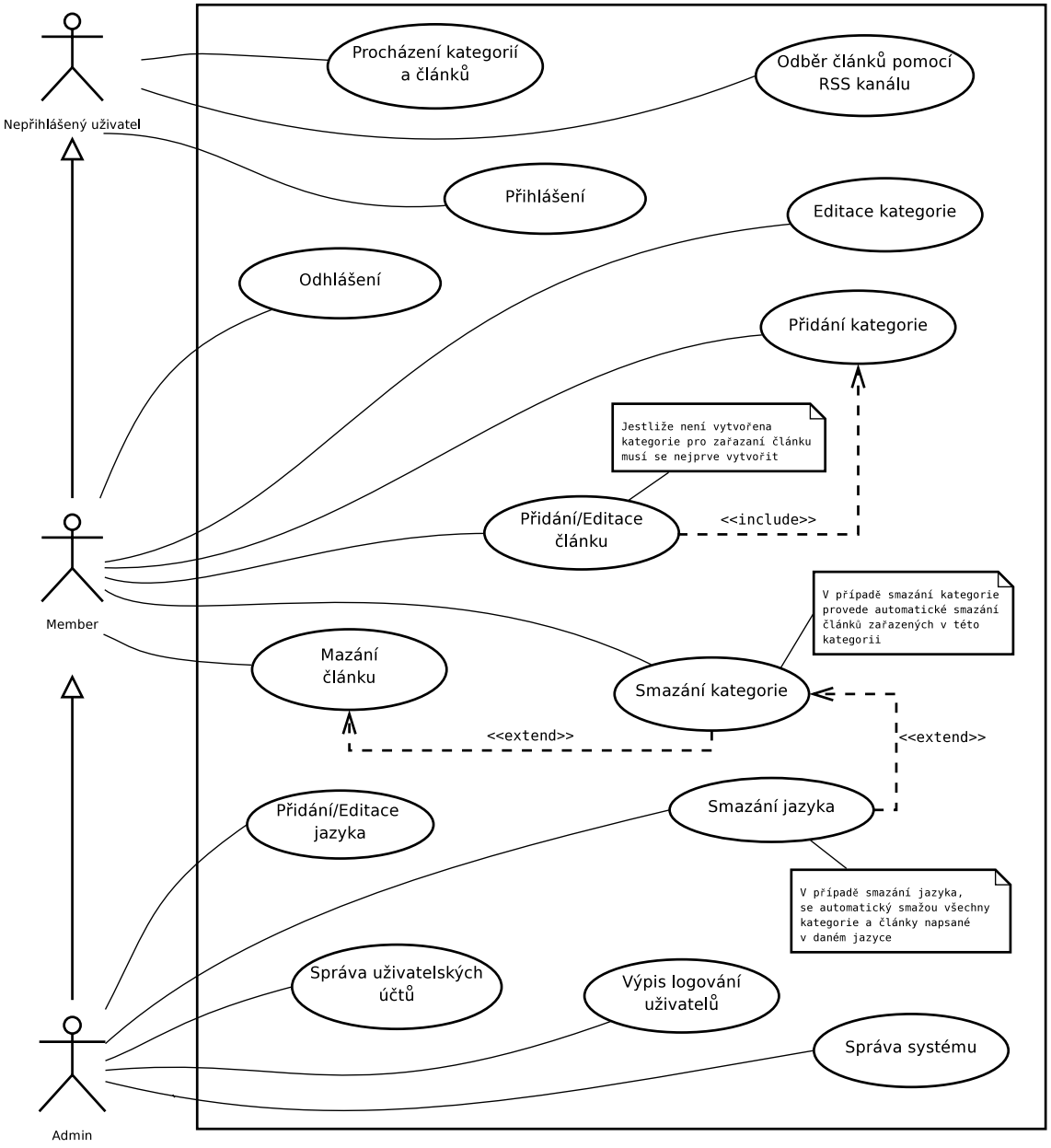
Stavové diagramy znázorňují stavy, ve kterých se systém nachází při běhu a reakcích uživatelů. STD diagramy obr. 8 a obr. 9 znázorňují systémové stavy při editaci kategorií a článků.

3.4 Systémové požadavky na implementaci a provoz

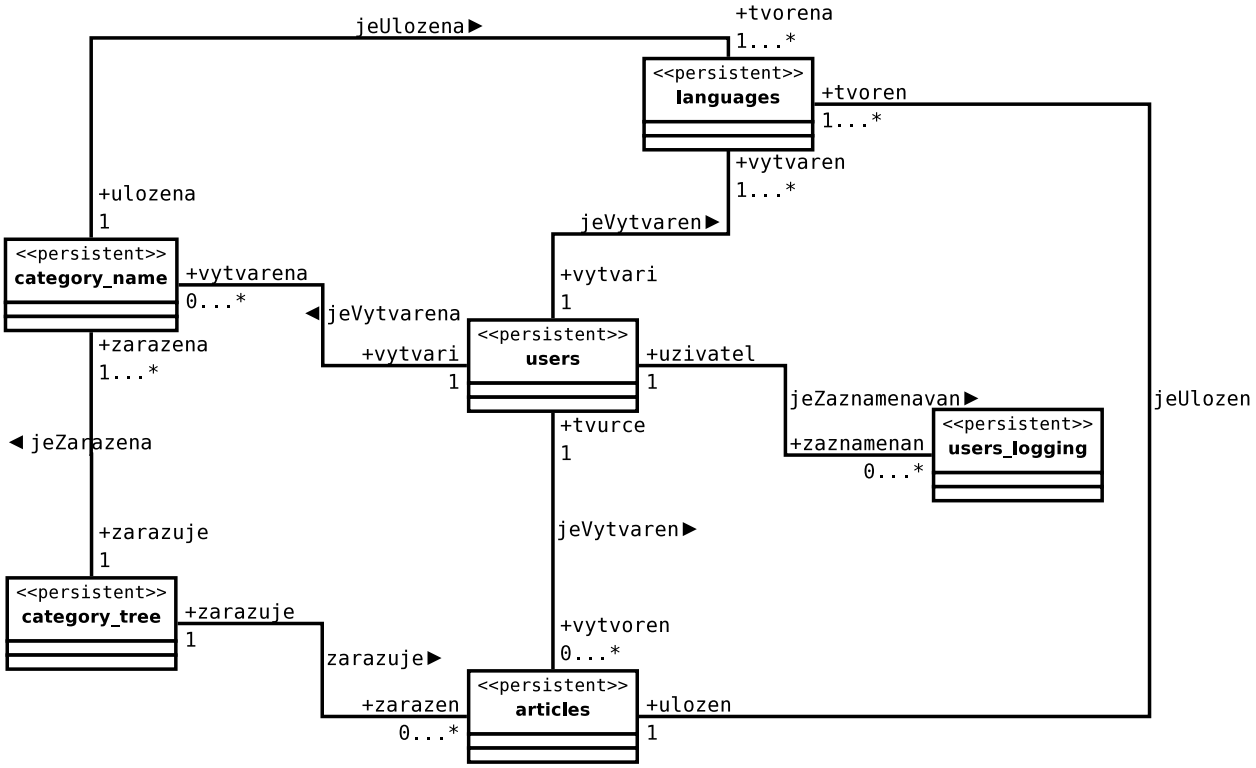
Analýza a systémové požadavky na implementaci a provoz jsou podrobně popsány v kapitole implementace.

3.5 Volba názvu systému

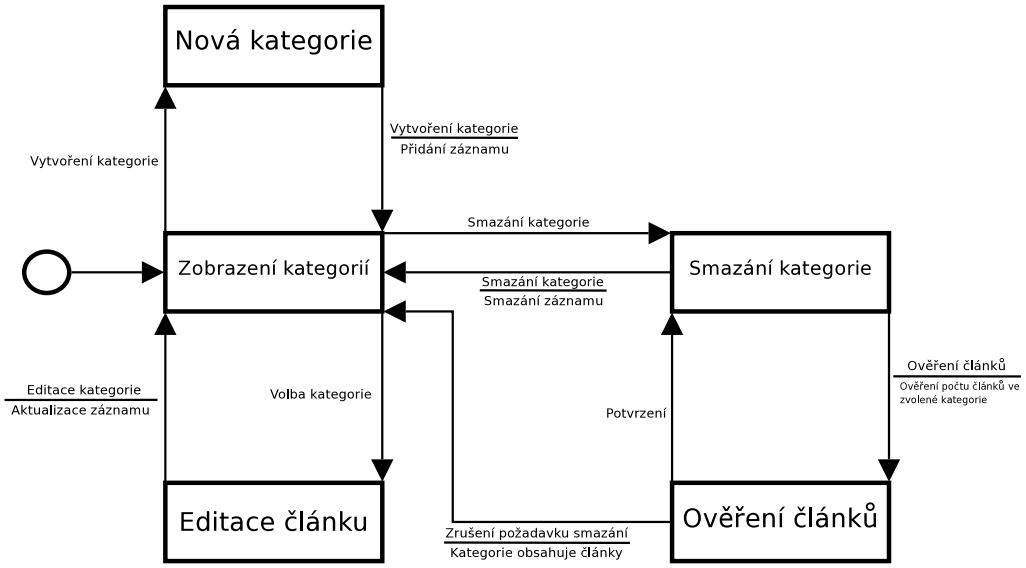
Jako název CMS redakčního systému byl zvolen „MicroCMS“. Tento název byl odvozen od vlastní webové domény `http://microstudio.cz`, tento název upřesňuje cílové nasazení této aplikace. Systém MicroCMS je jednoduchým CMS redakčním systémem určeného pro menší weby.



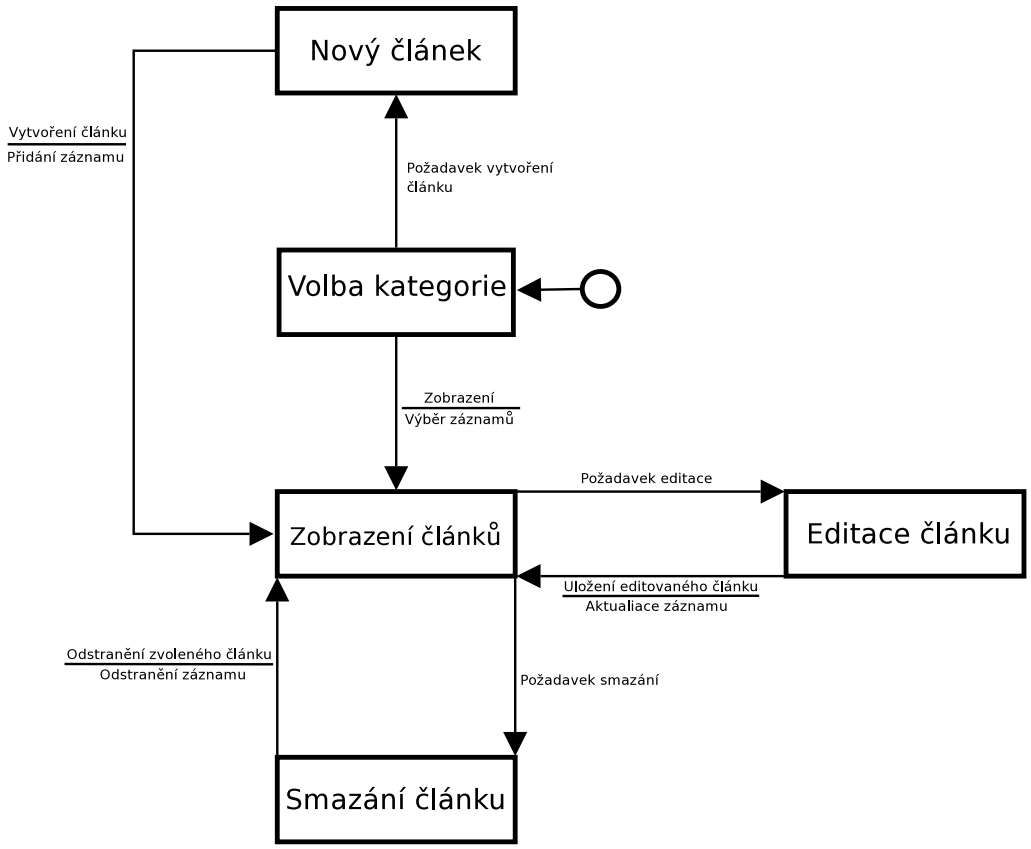
Obrázek 6: Diagram případů užití (Use Case Diagram)



Obrázek 7: Třídní diagram (Class Diagram)



Obrázek 8: Stavový diagram STD – správa kategorií



Obrázek 9: Stavový diagram STD – správa článků

3.6 Využití stromové struktry pro ukládání kategorií

Pro efektivní práci s ukládáním kategorií je vytvořena stromová struktura, která je uložena v tabulce *articles_category_tree*. Atributy tabulky *articles_category_tree* se skládají z hodnot idc, top, order a tree. Číselná hodnota *top* určuje úroveň zanoření (hlavní kategorie a podkategorií). Ve vlastním systému je navrženo 3-úrovňové menu, proto doména atributu *top* nabývá hodnot 0, 1, 2. Další atribut *id* je unikátní číslo kategorie a atribut *order* určuje pořadí jednotlivých kategorií, které se dá měnit. Z těchto hodnot se vytvoří textový řetězec, který vytváří stromovou strukturu. Stromový řetězec podkategorií vždy začíná hodnotou nadřazené kategorie a následuje vlastní hodnotou stromu. Tento způsob vytváření stromové struktury zjednodušuje SQL dotazy (pro výpis kategorie a jejich podkategorií stačí jednoduchý výběrový SQL dotaz s podmínkou LIKE (hodnota-stromu%)).

4 Implementace systému MicroCMS

Samotný vývoj systému MicroCMS je navržen tak, aby vytvořil jednoduché jádro webové aplikace, které bude snadno použitelné pro další projekty. Z funkcionálního hlediska využívá některé vlastnosti výše uvedených rozsáhlých CMS systémů. MicroCMS splňuje základní funkcionality CMS redakčního systému jako je správa kategorií a článků, správa uživatelů a jednoduchá obsluha pomocí vizuálního WYSIWYG editoru.

4.1 Systémové požadavky na implementaci a provoz

Hlavním cílem vývoje systému MicroCMS je použití opensource komponent. Jádro aplikace je postaveno na aplikačním rámci Zend Framework, jehož výhody a použité komponenty jsou popsány v následující kapitole. Vývoj aplikace je realizován ve skriptovacím jazyce PHP5 a jako databázový systém je primárně použit PostgreSQL[14] a MySQL[15].

4.1.1 Technologie pro vývoj a provoz systému MicroCMS

- webový server Apache nebo Microsoft IIS s podporou modulu rewrite pro přepisování URL adres.
- pro vývoj a běh byl zvolen jeden z nejrozšířenějších skriptovacích jazyků PHP5 (verze 5.2.6). PHP5 je rozšířen na většině webových serverů, může běžet na různými platformách a jeho využití je přímo koncipované pro vývoj webových aplikací.
- databázový systém PostgreSQL 8.3 a MySQL 5.0.41.
- aplikační rámec Zend Framework (aktuální verze 1.7.8).
- vývoj je realizován na platformě operačního systému Mac OS X (verze 10.5.6) s využitím aplikace MAMP PRO[16] určené pro vývoj webových aplikací.

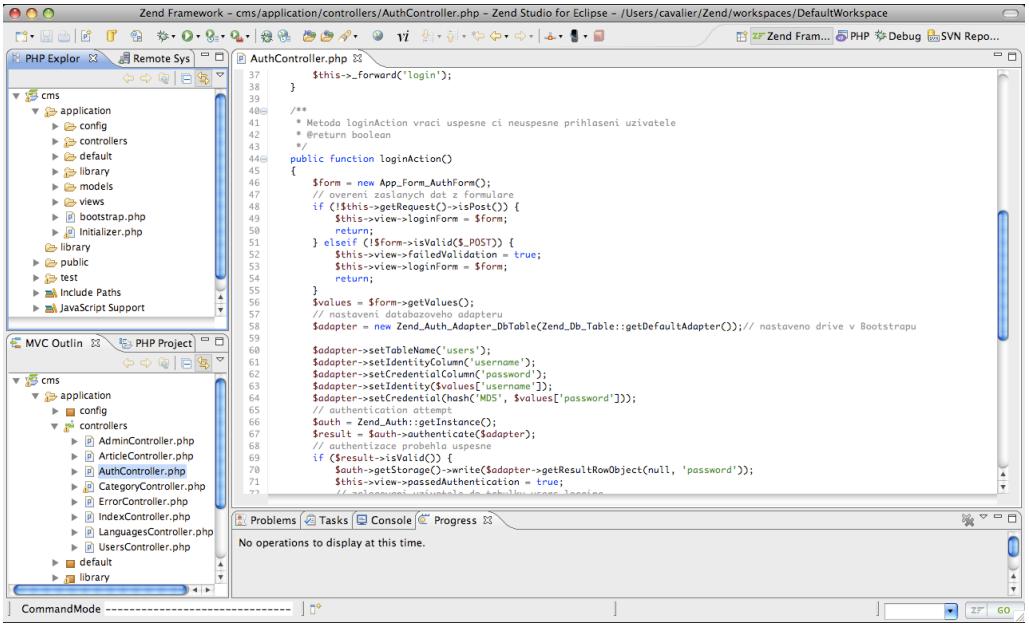
4.1.2 Volba vývojového prostředí

Vývoj je realizován ve vývojovém prostředí Zend Studio for Eclipse. Toto vývojové prostředí je vyvíjeno firmou Zend, která je tvůrcem Zend Frameworku a také vývojářem skriptovacího jazyka PHP (od verze 4). Ukázka vývojového prostředí je zobrazena na obrázku 10. Zend Studio for Eclipse je komerčním produktem, není tedy zdarma ke stažení jak je tomu například u vývojového prostředí Netbeans. Zend nabízí 30 denní zkušební verzi. Toto studio je vhodné pro vývoj aplikací v jazyce PHP, protože poskytuje

[14] PostgreSQL – je objektově-relační opensource databázový systém. Jedná se o velmi rozsáhlý SŘBD systém, který je vhodný i pro enterprise podniková řešení. Systém je dostupný pro většinu platforem operačních systémů jako jsou Windows, Unix, Linux a další.

[15] MySQL – je dalším populárním SŘBD systémem. Jedná se o jeden z nejrozšířenějších databázových systémů, které se využívají v prosředí internetu.

[16] MAMP PRO – soubor programů a skriptovacích jazyků určených pro vývoj webových aplikací na platformě Mac (zkratka znamená Macintosh, Apache, MySQL a PHP).
Oficiální stránka projektu `http://www.mamp.info/`



Obrázek 10: Vývojové prostředí Zend Studio for Eclipse

mnoho funkcionalit, dokumentace a usnadnění vývoje nad aplikačním rámcem Zend Framework.

4.2 Návrh struktury aplikace

V realizaci vlastního systému jsou dodrženy moderní trendy programování a programovacích technik. Programovací jazyk PHP5 již nabízí kompletní podporu objektově-orientovaného programovaní.

Dalším trendem vývoje webových aplikací je využití 3-vrstvé architektury, která má následující strukturu:

- **prezentační** – zajišťuje interakce s uživatelem, grafický výstup, zobrazování vstupních a výstupních formulářů.
- **funkční** – zajišťuje funkční část aplikace, ověřuje správnost vstupních informací, filtraci, kontrolu, výpočty apod.
- **datová** – prezentuje data, která mohou být uložena v databázovém úložišti či souboru.

Hlavní důraz při vývoji je kladen na funkční (aplikační) a prezentační vrsvu. Pro dodržení této architektury je využíváno komponent Zend Frameworku.

4.3 Aplikační rámec neboli framework

Framework je programový základ, nad kterým se vyvíjejí aplikace. Framework není sám o sobě hotovým programem, jedná se o skupinu hotových komponent a rozhraní, na kterých se samotný systém vyvíjí. Některé frameworky jsou koncipovány tak, aby programátor při vývoji aplikace dodržoval objektově-orientované programování a návrhové vzory pro vývoj aplikace. Jedním z nejčastěji využívaných návrhových vzorů pro vývoj webových aplikací je MVC neboli Model-View-Controller (Model-Pohled-Řadič). Aplikační frameworky dále poskytují množství API rozhraní. Zend Framework například poskytuje možnost propojení vlastní aplikace se službami společností jako jsou Google, Amazon, Delicious, Flickr, „Yahoo!“ a další. Díky těmto možnostem se může snadno přistupovat ke službám jako jsou mapy Google Maps, fotogalerie Google Picasa či Flickr, obchodní portál Amazon a další. Samotný framework také hlídá průběh programování, tzn. upozorňuje na možné výskyty chyb a dodržování standardů programování, aby splňovali PHP E_STRICT. Další výhody frameworku budou vysvětleny v následujících kapitolách při detailním popisu Zend Frameworku.

4.4 Návrhový vzor Model-View-Controller

Návrhový vzor Model-View-Controller se využívá proto, aby rozdělil strukturu aplikace na 3-vrstvou architekturu. Prezentační vrstvu tvoří View, funkční (aplikační) vrstvu tvoří Controller a pro práci s daty se využívá Model. Výhodou této 3-vrstvé architektury je snadná modifikace a nezávislost jedné vrstvy na druhé. Jako příklad využití může být změna ve View, kdy se změní technologie zobrazování, přičemž se nemusí zasahovat do logické a datové části aplikace.

Popis funkcí jednolivých částí návrhového vzoru MVC:

4.4.1 Model

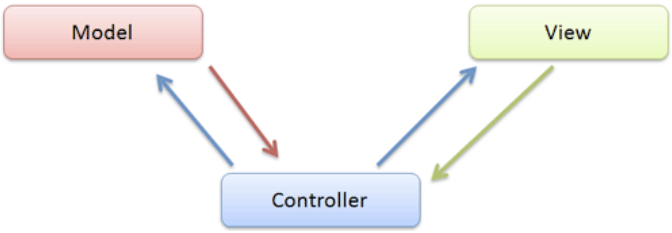
Model reprezentuje data, se kterými aplikace pracuje. Jedná se o specifickou reprezentaci informací, nad kterými se provádí operace. V modelu se programuje logika, která pracuje s daty jako jsou výpočty, různá zjištění dostupných dat, ověření apod.

4.4.2 View

View neboli pohled reprezentuje data tvořená modelem do podoby vhodné pro zobrazení koncovému uživateli. Toto zobrazení zajišťuje uživatelské prostředí (v protředí webových aplikací jsou to formuláře a (X)HTML tagy). Vstupní formuláře zajišťují vkládání uživatelských dat (vstupů), které jsou po odeslání formuláře předány kontroleru.

4.4.3 Controller

Zpracovává a reaguje na události vyvolané uživatelem. V controlleru neboli řadiči je naprogramována hlavní logika aplikace, která provádí operace nad daty reprezentova-



Obrázek 11: Schéma návrhového vzoru Model-View-Controller

nými modelem nebo reaguje na uživatelské událost jako jsou stisknutí tlačítka a odeslání formulářových dat.

4.5 Volba aplikačního rámce Zend Framework

Exituje řada aplikačních rámců pro vývoj aplikací v jazyce PHP, proto bylo nutné některý z těchto frameworků vybrat. Frameworky se liší množstvím dostupných knihoven, některé jsou napsány v jazyce PHP4. V přehledové tabulce stávajících frameworků byl nejzajímavější framework českého vývojáře Nette Framework a rozsáhlý Zend Framework. Po delším zvažování byl volen Zend Framework (dále ZF), protože nabízí větší možnosti a množství dostupných komponent a především je vyvíjen společností Zend, která je vývojářem jazyka PHP. Další výhodou je rozsáhlá základna vývojářů, zatímco Nette Framework naprogramoval jediný programátor. Na vývoji Zend Frameworku se podílí rozsáhlý vývojový tým firmy Zend. Dalšími výhodami jsou přísné vývojové postupy. Zend Framework se zaručuje několika vývojovými cíly. Jedním z cílů vývoje je zpětná kompatibilita s předchozími verzemi, kdy v případě vydání oprav bezpečnostních chyb stačí pozměnit systém knihoven samotného frameworku a nemusí se zasahovat do struktury aplikace, která tento framework pro svůj běh využívá. Pokud se ve frameworku provedou zásadní změny, jako jsou změny v názvech tříd či metod, Zend oznámí tyto zásadní změny ihned s vydanou verzí, aby se mohla upravit konkrétní část aplikace využívající tyto třídy.

4.5.0.1 Výhody volby ZF

- je založen na jednoduchosti a využití objektově-orientovaných praktik programování zajišťující kvalitu a trend vývoje webových aplikací.
- je kompletně napsán v jazyce PHP5.
- vývoj je zaměřen na zajištění spolehlivosti, vysoké míry bezpečnosti, využití moderních technologií a poskytuje připojení pro webové služby skrze rozhraní API předních dodavatelů jako je Google, Yahoo!, Amazon a Flickr.
- framework je distribuován a šířen pod „přátelskou“ licenční smlouvou New BSD, která je tou nejsvobodnější licencí v oblasti svobodného software. S využitím této

licence se mohou jakkoliv upravovat stávající kódy, distribuovat a vývojáři nejsou nijak licenčně omezeni. Jedinou podmínkou této licence je uvedení autora softwarové aplikace a informace o licenci.

- framework vyžívá výše zmíněné metody 3-vrstvé architektury s využitím návrhového vzoru Model-View-Controller.
- obsahuje velké množství dostupných komponent.
- komponenty frameworku nejsou mezi sebou nijak závislé, komponenty které nevyužívám mohu z knihoven odstranit a tím minimalizovat velikost aplikace.
- podpora rozhraní pro multidatabázový přístup. Rozhraní podporuje nejznámější SŘBD systémy jako jsou MySQL, PostgreSQL, MS SQL Server, Oracle, SQLite a další.
- všechny komponenty a knihovny jsou plně testovány, aby byla zajištěna jejich stabilita a bezproblémový chod.

4.5.0.2 Nevýhody

- mezi nevýhody patří zatím malá komunita vývojářů v ČR. Většina manuálů, hotových řešení a užitečných informací je k nalezení na anglických webech, proto není problém se zjištěním informací a užitečných rad. V ČR již existuje český web s komunitním fórem[17].
- první začátky programování s frameworkem jsou zdlouhavé. Jde o pochopení základů a nového způsobu programování založeného na aplikačním frameworku a 3-vrstvé architektuře. Po zvládnutí těchto základů byl samotný vývoj aplikace velmi rychlý a efektivní.
- poslední nevýhodou je pomalejší běh, tato informace byla zveřejněna z provedeného testu nejznámějších PHP Frameworků[18]. S využitím eAcceleratoru je tato studie zcela zanedbatelná, v testu se ukázalo zrychlení provedených kroků u většiny frameworků na rychlost okolo 40ms.

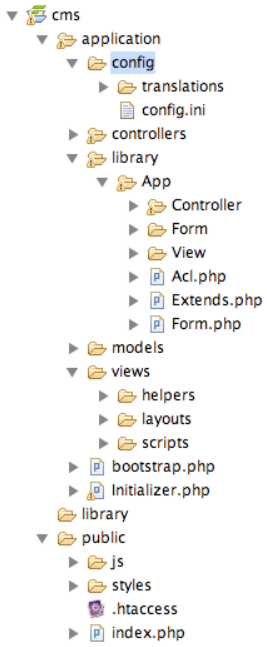
[17] `http://www.zendframework.cz/`
[18] `http://www.root.cz/serialy/velky-test-php-frameworku/`

4.6 Základní rozvržení struktury aplikace

Pro zvýšení bezpečnosti je webová aplikace rozvržena do následující struktury. Z webu je pouze přístupná složka *public*, ostatní složky mají přístup zvenčí chráněny.

- **application** – hlavní aplikace
  - **configuration** – konfigurační soubory (volba z konfiguračních textových souborů INI nebo formátu XML)
  - **controller** – controllery – zajišťují funkční část aplikace
  - **model** – modely pro práci s daty
  - **view** – pohledy, reprezentace dat
    - **filters** – filtry
    - **helpers** – pomocníci – jsou to skripty pomocí kterých lze na stránku napsat jednoduché funkční prvky (např. zobrazování aktuálně přihlášeného uživatele apod.)
    - **layouts** – vzhled aplikace
    - **scripts** – obsahuje složky s pojmenovanými názvy podle řadičů, v každé složce jsou uloženy phtml soubory se skripty, jakým způsobem se má daná akce řadiče zobrazovat
  - **bootstrap.php** – nejdůležitější „jádro“ aplikace, jedná se o hlavní soubor, ve kterém se nastavuje aplikační prostředí, cesty, routování a spousta dalších nastavení
- **library** – složka knihoven – v této složce je uložen samotný Zend Framework, mohou se zde ovšem uložit i další knihovny či frameworky
  - **Zend** – Zend Framework
- **public** – veřejná složka
  - **images** – obrázky
  - **scripts** – skripty jazyka JavaScript
  - **styles** – kaskádové styly
  - **.htaccess** – soubor pro rewrite mód, zajišťující zabezpečení a povolení souborů a složek, které mohou být klientem zobrazeny
  - **index.php** – spoustěcí PHP soubor

Tento způsob rozvržení aplikace je jen doporučený. U složitějších aplikací se používá modulární rozvržení, kde každý modul jako je např. blog, e-shop, katalog je rozdělen do zvláštních složek, které obsahují vlastní složky s kontrolery, pohledy a modely. Rozvržení struktury systému MicroCMS je rozděleno viz obrázek 12.



Obrázek 12: Rozvržení struktury aplikace systému MicroCMS

4.7 Tvorba layoutu a uživatelského prostředí

Pro layout neboli vzhled systému je zvolena šablona z projektu freeCSStemplates.org[19]. V šabloně byly upraveny stávající kaskádové styly, přepsány části struktury (X)HTML s přidáním několika grafických doplňků. Cílem projektů zabývajících se CSS šablonami je vytvoření pevné (X)HTML struktury, jejíž vzhled se mění jen pomocí CSS kaskádových stylů. Mezi tyto projekty patří freeCSStemplates.org či CSS Zen Garden[20]. Výhodou XHTML s využitím CSS je jednoduchost hlavní XHTML struktury aplikace a vysoká flexibilita změny vzhledu pomocí CSS kaskádových stylů. Uživatelské prostředí „spring" systému MicroCMS je na obrázku 13.

4.8 Ukázky výsledného řešení

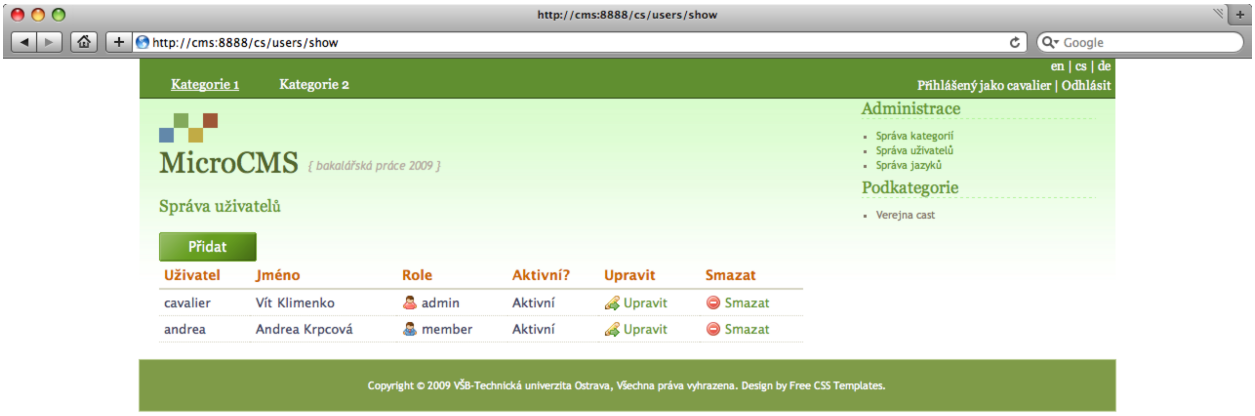
4.8.1 Zobrazení článku

Ukázka zobrazení článku zařazeného v podkategorii 3. úrovně. Viz obrázek 14.

4.8.2 Editace článku pomocí WYSIWYG editoru

Editaci obsahu článků zajišťuje vizuální WYSIWYG editor TinyMCE[21]. Na obrázku 15 je zobrazena editace článku „Kalous ušatý".

[19] `http://www.freecsstemplates.org/`
[20] `http://www.csszengarden.com/`
[21] Oficiální stránka WYSIWYG editoru TinyMCE `http://tinymce.moxiecode.com/`.



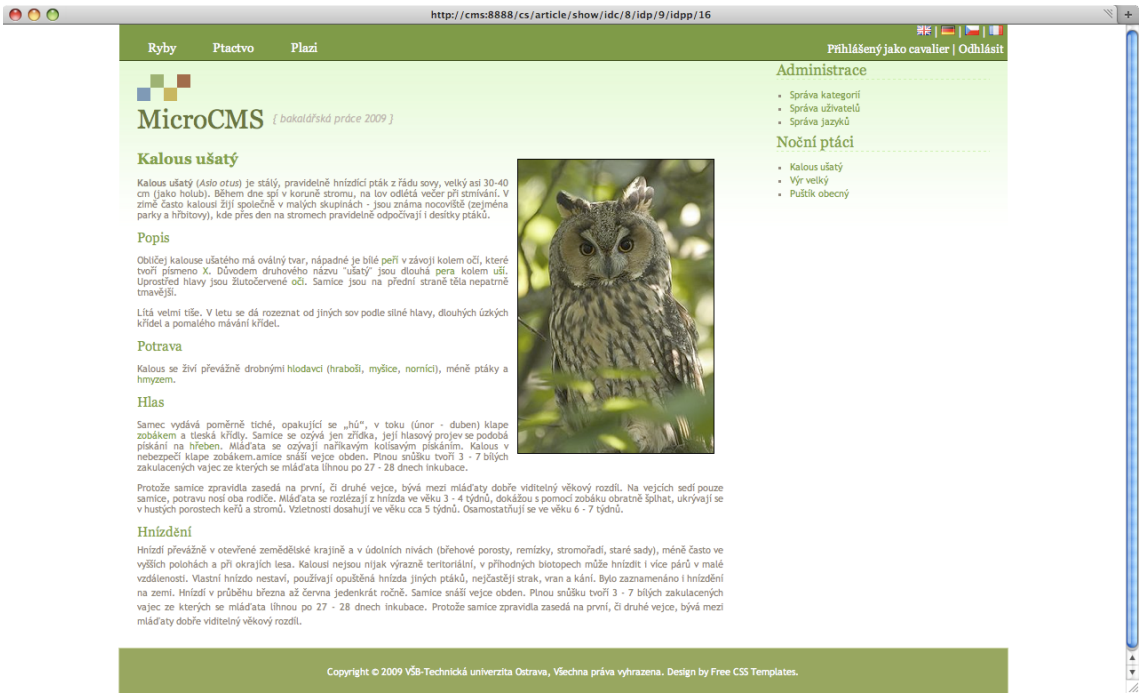
Obrázek 13: Uživatelské prostředí „spring“ systému MicroCMS

4.8.3 Správa kategorií

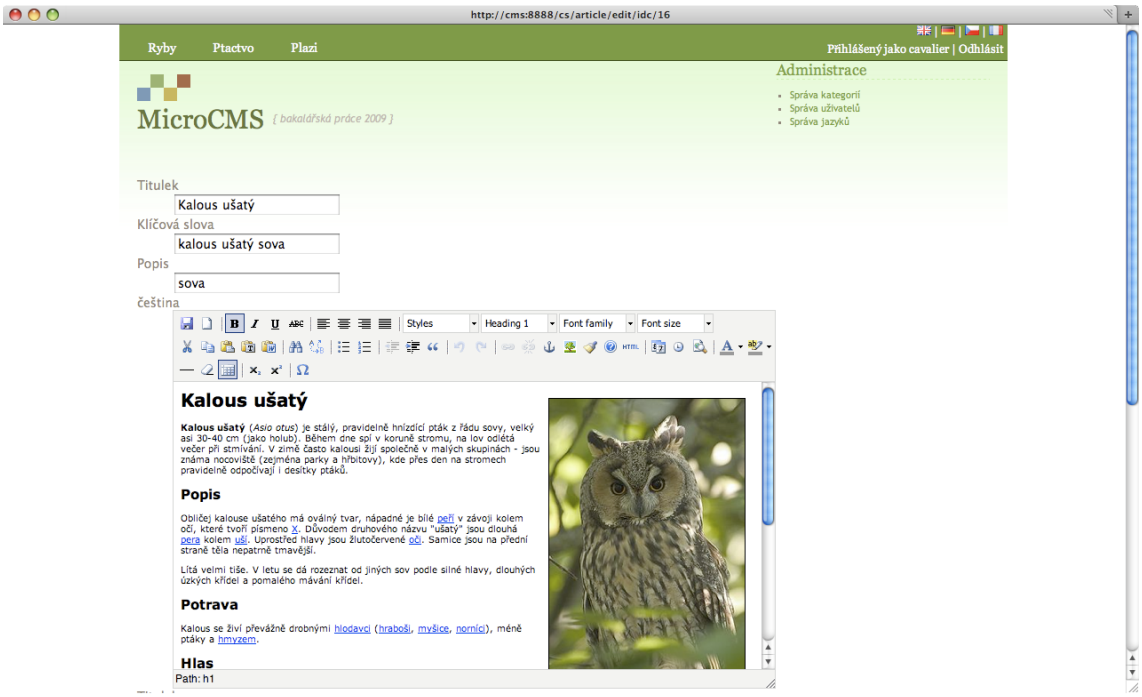
Správa kategorií vypisuje seznam vytvořených kategorií v pořadí určeného stromovou strukturou. Viz obrázek 16.

4.8.4 Správa uživatelských účtů

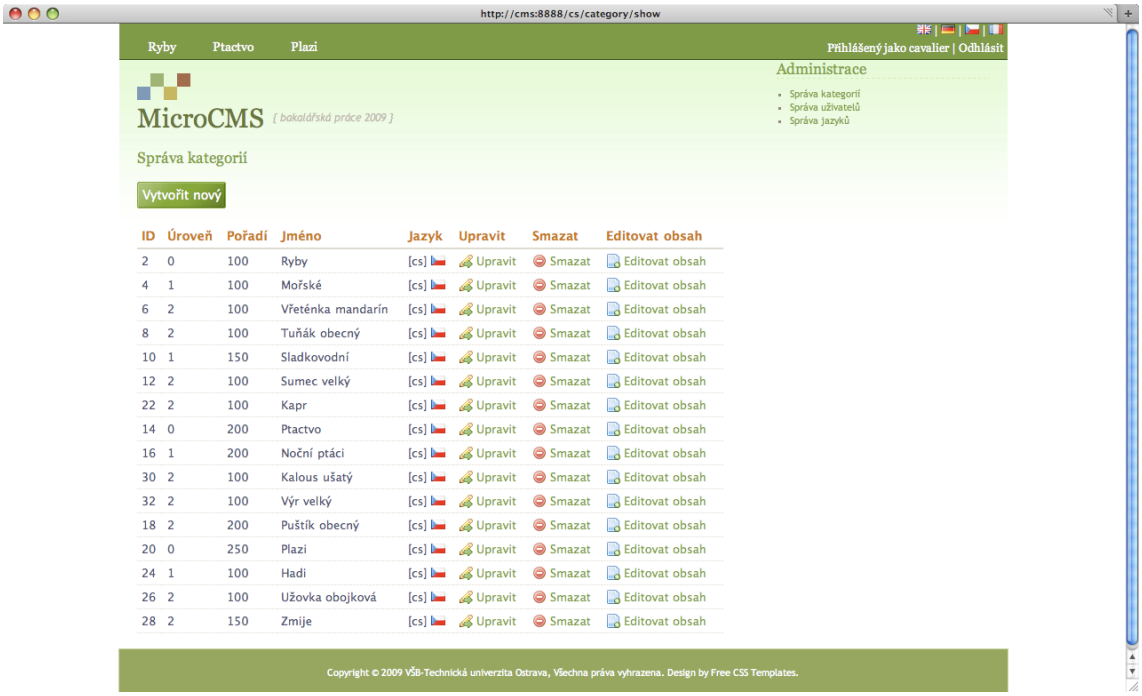
Správa uživatelských účtů vypisuje seznam uživatelů a tabulku s údaji o posledním přihlášení a celkové době přihlášení v systému. Viz obrázek 17.



Obrázek 14: Zobrazení článku



Obrázek 15: Administrační část - editace článku pomocí WYSIWYG editoru



Obrázek 16: Administrační část - správa kategorií

Obrázek 17: Administrační část - správa uživatelských účtů

4.9 Přehled knihoven využitých v systému

Autentizace a autorizace

- **Zend_Auth** – rozhraní pro autentizaci, obsahující adaptéry pro provedení autentizace. Adaptér Zend_Auth podporuje několik autentizačních metod, od základní HTTP autentizace podporované webovým serverem, přes autentizaci s údaji uloženými v databázovém úložišti (využito v systému), dalšími metodami jsou autentizace přes LDAP, OpenID.
- **Zend_Acl** – poskytuje možnost autorizace neboli řízení přidělování oprávnění k objektům a akcím. V systému MicroCMS je využita metoda zdrojů a rolí. Zdroj znamená objekt, ke kterému je kontrolován přístup. Rolí se rozumí objekt, který žádá o přístup ke zdroji.

Konfigurace

- **Zend_Config** – je navržen tak, aby se zjednodušil přístup a využití konfiguračních dat v rámci celé aplikace. Knihovna nabízí vytvoření konfiguračních souborů v textovém formátu INI a XML. Data jsou v konfiguračních souborech rozdělena do oblastí a samotné oblasti se mohou mezi sebou dědit.

Model-View-Controller

- **Zend_Controller** – nejdůležitější knihovna frameworku, zajišťuje běh a rozvržení aplikace pomocí návrhového vzoru MVC.

Datum a čas

- **Zend_Date** – zobrazení datumů a času.

Databázový přístup

- **Zend_Db** – poskytuje rozhraní pro přístup k různým SŘBD systémům a má v sobě integrovaný systém pro tvorbu databázových dotazů[22].

Hledání chyb

- **Zend_Debug** – zajišťuje jednoduchou techniku ladění aplikace. Slouží k usnadnění vývoje aplikace.

Filtrace vstupních dat

- **Zend_Filter** – poskytuje jednoduchý filtrační mechanismus, který je využíván při ověřování vstupních dat. Filtry zajišťují, aby uživatel vkládal do vstupních formulářů správná data a ověření správnosti formátu.

[22] V každém databázovém systému se liší struktura dotazovacího jazyka a také seznam datových typů, proto je v knihovně integrován modul pro tvorbu dotazů, aby tyto rozdíly sjednotil.

Tvorba formulářů

- **Zend_Form** – zjednodušuje tvorbu (X)HTML formulářů, zajišťuje validaci dat s kombinací Zend_Filter, sdružuje (X)HTML elementy a zajišťuje dekorace – volbu rozmístění formulářů.

Layout – vzhled aplikace

- **Zend_Layout** – vytváří šablonu, ve které se aplikace zobrazuje (vzhled stránky).

Registry

- **Zend_Registry** – je mechanismus k používání globálních proměnných, které jsou dostupné a viditelné v rámci celé aplikace.

Lokalizace

- **Zend_Locale** – knihovna pro zajištění aplikační lokalizace tzn. zobrazování data, času, jmen, příjmení, měn a dalších informací ve správném formátu národního prostředí.

Správa sezení

- **Zend_Session** – zajišťuje správu sezení pomocí které se zajistí odstranění bezstavovosti HTTP protokolu.

Vícejazyčný systém

- **Zend_Translate** – tvorba vícejazyčných systémů, knihovna zajistí zobrazování přeložených textů pro zvolený jazyk. Překlady systému jsou uloženy v textových souborech. Knihovna podporuje řadu formátů pro ukládání překladů jako jsou XML, INI, CSV, Gettext, ukládání v polích a další.

Pohledy

- **Zend_View** – doplňuje Zend_Controller pro využití 3-vrstvé architektury aplikace.

4.9.1 Seznam dalších užitečných knihoven

- **Zend_Cache** – zajištění kešování za chodu aplikace.
- **Zend_Captcha** – vytváření obrázků Captcha, které slouží k zamezení spamových útokům. Setkáváme se s ní u sms-bran, registračních formulářů atd.
- **Zend_Config_Writer** – vytváření konfiguračních souborů.
- **Zend_Currency** – práce s měnami.

- **Zend_Log** – využívá se k ukládání logů a k ladění aplikace při vývoji.
- **Zend_Mail** – rozsáhlá knihovna pro práci s emailovými zprávami (odesílání a čtení zpráv).
- **Zend_Navigation** – využívá se pro tvorbu menu, map stránek nebo pro účely navigace.
- **Zend_Paginator** – vytváření stránkování.
- **Zend_Pdf** – vytváření a úprava PDF dokumentů.
- **Zend_Search_Lucene** – pracuje s dokumenty jako s atomickými objekty pro indexování. Tato knihovna se využívá pro vyhledávání informací v aplikaci.
- **Zend_Service_*** – zajišťuje možnost připojení na rozhraní API různých služeb jako je Google, Amazon, Flickr, Yahoo a další.

Toto není zdaleka kompletní seznam knihoven Zend Frameworku. Tento framework je velmi rozsáhlý a poskytuje velké množství užitečných komponent. Více informací je k nalezení v oficiálním manuálu `http://framework.zend.com/`.

4.10 Konfigurační soubory

Pro konfiguraci ve vlastní implementaci se využívá knihovny Zend_Config, ze dvou typů zápisů souborů byly zvoleny soubory INI, které jsou lépe čitelnější. Pro konfigurační soubory je vytvořena složka *config*, ve které je umístěn hlavní konfigurační soubor *config.ini*. V ukázce je uveden malý příklad zápisu konfiguračních hodnot. Jedná se o konfigurační hodnoty pro nastavení databázového adaptéru (systém PostgreSQL, běžící na lokálním počítači, přihlašovací jméno s hesly, pro název databáze je zvoleno jméno cms). Viz výpis 1.

```
; volba adapteru
db.adapter = PDO_PGSQL

; domenove jmeno ci IP adresa databazoveho stroje
db.params.host = localhost

; prihlasovaci  jmeno
db.params.username = uzivatelske_jmeno

; heslo
db.params.password = heslo

; nazev databaze
db.params.dbname = cms

; volba layoutu
layout = spring
```

Výpis 1: Konfigurační soubor INI

4.11 Autentizace

Pro zajištění autentizace neboli ověřování totožnosti uživatelů je vytvořen autentizační kontroler s názvem *AuthController*. V tomto kontroleru jsou dvě důležité metody, první metoda *loginAction* slouží k ověření totožnosti přihlašujícího se uživatele s využitím autentizačního adaptéru *Zend_Auth_Adapter_DbTable*. Kontroler ověří zadané formulářové hodnoty uživatelského jména a hesla s hodnotami uloženými v tabulce *users*. V případě úspěšného ověření zapíše logovací informace do tabulky *users_logging*, poté je uživatel do systému přihlášen. V případě neúspěchu oznámí, že ověření neproběhlo úspěšně (chybné uživatelské jméno či heslo). Ve výpise 2 je zobrazena část zdrojového kódu metody *loginAction*.

```
// po uspesnem ziskani formularovych hodnot vytvorime autentizacni adapter
$adapter = new Zend_Auth_Adapter_DbTable(Zend_Db_Table::getDefaultAdapter());
$adapter->setTableName('users'); // volba DB tabulky
$adapter->setIdentityColumn('username'); // nazev sloupce s uzivatelskymi jmeny
$adapter->setCredentialColumn('password'); // hesla
$adapter->setIdentity($values['username']); // ziskani hodnot z prihlasovaciho formulare
$adapter->setCredential(hash('MD5', $values['password']));

// provedeni autentizace
$auth = Zend_Auth::getInstance();
$result = $auth->authenticate($adapter);

// autentizace probehla uspesne
if ($result->isValid()) {
    $auth->getStorage()->write($adapter->getResultRowObject(null, 'password'));
    $this->view->passedAuthentication = true;

    // zalogovani prihlaseni do tabulky users_logging
    $users = new Users();
    $user = $users->fetchRow("username='" . $values['username'] . "'");
    $userId = $user->id;
    $data = array('idu' => $userId, 'date_login' => date('Y-m-d H:i:s'), 'ip' => $_SERVER['
        REMOTE_ADDR']);
    $this->_db->insert('users_logging', $data);
    $this->_namespace->saveId = $this->_db->lastInsertId('users_logging', 'id');
    $this->_namespace->role = $user->role;

    // po uspesnem prihlaseni presmerujeme na pozadovany skript
    $this->_redirect('/' . $this->_lang . '/admin/');
} else {
    // chyba! autentizace neprobehla uspesne
    $this->view->failedAuthentication = true; // zobrazime chybu prihlaseni
    $this->view->loginForm = $form; // zobrazime znovu prihlasovaci formular
}
```

Výpis 2: Metoda loginAction kontroleru AuthController

Systémové odhlášení pomocí metody *logoutAction* je velmi jednoduché. Stačí zavolat instanci *Zend_Auth* a její metodu *clearIdentity*, tím se provede samotné odhlášení (zrušení relace). Při přihlášení je do sezení uložena aktuální hodnota ID přihlášeného uživatele,

kdyby tato hodnota byla ztracena, nepodařilo by se při odhlášení zapsat čas odhlášení pro aktuálně přihlášeného uživatele. Viz výpis 3.

```
// odhlaseni ze systemu
Zend_Auth::getInstance()->clearIdentity();

$saveId = $this->_namespace->saveId; // zjisteni hodnoty ID
if (isset($saveId)) { // zapiseme cas odhlaseni do tabulky users_logging
    $this->_db->update('users_logging', array('date_logout' => date('Y-m-d H:i:s')), 'id=' .
        $saveId);
    unset($this->_namespace->saveId); // zrusime stavaji promenne
    unset($this->_namespace->role);
}
$redirector = $this->_helper->redirector; // presmerujeme
$redirector->gotoRouteAndExit(array('lang' => $this->_lang), null, true);
```

Výpis 3: Metoda logoutAction kontroleru AuthController

4.12 Ukázka modelu

V následujícím výpise 4 je zobrazen zdrojový kód řešení modelu *Users*, ve kterém je obsažena jednoduchá metoda pro zjistění počtu uživatelů v systému.

```
class Users extends Zend_Db_Table_Abstract // trida Users zdedi vlastnosti z tridy
    Zend_Db_Table_Abstract
{
    protected $_name = 'users'; // nazev tabulky

    public function countUsers() // metoda pro zjisteni poctu uzivatelu
    {
        $select = $this->select(); // zavolani jednoducheho databazoveho dotazu
        $select->from($this->_name, 'COUNT(*) AS num');
        return $this->fetchRow($select)->num;
    }

    public function __toString () // metoda __toString pro vraceni nazvu tabulky
    {
        return $this->_name;
    }
}
```

Výpis 4: Ukázka jednoduchého modelu pro práci s daty nad tabulkou users

4.13 Vytvoření akce pro získání dat uložených v DB

Pro ukázku jednoduché akce získání dat je vypsána část zdrojového kódu pro vytvoření metody *showAction* v kontroleru *UsersController*. Tato akce získá data uložená v databázové tabulce *users*, která jsou reprezentována modelem *Users*. Díky Zend Frameworku se tyto databázové dotazy a práce se SŘBD mnohonásobně zjednodušší. Pro získání dat stačí vytvořit instanci modelu *Users* a zavolat dotazovací metodu. Viz výpis 5.

```
public function showAction()
{
    $table = new Users(); // vytvoreni instance modelu Users
    $select = $table->select()->order('id'); // zavolani dotazovaci metody (seradime podle
        parametru ID)
    $rows = $table->fetchAll($select); // provedeme ziskani dat
    $this->view->users = $rows->toArray(); // ziskana data predame pohledu View na zobrazeni
}
```

Výpis 5: Vytvoření akce pro získání dat uložených v databázi

4.14 Zobrazení získaných dat v pohledu

Pro každý kontroler se vytvoří pro *view* zvláštní složka s názvem kontroleru. Každá akce kontroleru má vytvořený vlastní soubor s příponou *phtml*. V předchozí ukázce byla získána data uložená v databázové tabulce. Ve výpise 6 je předvedeno vytvoření pohledu, jak tyto získaná data interpretovat uživateli.

```
<?php
if (count($this->users) > 0) : // pokud se pocet ziskanych dat nerovna nule vypisi tabulku
    ?><D-Y>
<table width="100%" cellspacing="0" cellpadding="0" id="users"> // vykresleni tabulky s hlavickou
    <tr>
        <th><?= $this->translate('Username') ?></th> // metoda translate zajistuje vypis retezce
            ve zvolenem jazyce systemu
        <th><?= $this->translate('Real name') ?></th>
        <th><?= $this->translate('Role') ?></th>
        <th><?= $this->translate('Active') ?>?</th>
        <th><?= $this->translate('Edit') ?></th>
        <th><?= $this->translate('Delete') ?></th>
    </tr>
// v cyklu ziskam a vypisi data, ktera byla predana pomoci pole z akce showAction v
    UsersControlleru
<?php foreach ($this->users as $user) : ?>
    <tr class="c">
        <td><?= $user['username']?></td> // reprezentace uzivatelskeho jmena
        <td><?= $user['name'] . ' ' . $user['surname']?></td> // jmeno
        <td><span class="<?=$user['role']?>"><?=$user['role']?></span></td> // role
        <td><?= ($user['active'] == true ? $this->translate('Active') : $this->translate('Blocked'))
            ?></td> // vypsani zda je uzivatel blokovany
        // vytvoreni odkazu pro editaci
        <td><a href="<?= $this->baseUrl() . '/users/edit/id/' . $user['id']?>" class="edit"><?=
            $this->translate('Edit')?></a></td>
        <td><a href="<?= $this->baseUrl() . '/users/delete/id/' . $user['id'] ?>" class="delete"
        onclick="return confirm('<?=$this->translate('Are you sure you want to delete')?>?')">
        <?=$this->translate('Delete')?></a></td> // vytvoreni odkazu pro mazani
    </tr>
<?php endforeach; ?>
</table>
<?php endif; ?>
```

Výpis 6: Zobrazení získaných dat v pohledu

5 Závěr

Cílem této práce bylo vypracování přehledové studie, zjištění základních vlastností a vlastní implementace CMS redakčního systému.

V úvodu práce byly popsány obecné vlastnosti informačních webových systémů a jejich základní funkcionalita. Pro vypracování přehledové studie stávajících hotových řešení byly vybrány nejznámější a nejrozšířenějších CMS redakční systémy (Wiki, „Joomla!“, Drupal a další). Vlastní realizace byla inspirována několika stávajícími opensource RS.

MicroCMS splňuje základní funkcionalitu RS jako je správa kategorií a článků, správa uživatelů a zajištění snadné obsluhy. Podmínkou zadání a cílem vlastního řešení bylo zajištění vícevrstvé architektury systému, přívětivé uživatelské rozhraní, flexibilita prezentační části a zajištění vývoje systému na stávajících opensource komponentách. Jako základní stavební kámen celého systému byl zvolen Zend Framework, který značně usnadnil vývoj a zajistil splnění výše zmíněných podmínek. Pro vytvoření uživatelského rozhraní byly použity technologie (X)HTML a CSS s využitím CSS šablon. Programová část systému byla řešena ve skriptovacím jazyce PHP5 s využitím databázového systému PostgreSQL a MySQL.

Vlastní systém se zatím nemůže srovnávat se stávajícími rozsáhlými RS, samotné jádro vlastního systému ovšem zajišťuje další možnosti rozšíření, ve kterých lze ve vývoji nadále pokračovat. Jedním z hlavních cílů je úprava systému do modulární struktury a vytvoření autorizačních rolí a zdrojů uložených v databázovém systému. Stávající systém je vhodný pro nasazení na menším webu.

6 Literatura

[1] Jesus Castagnetto, Harish Rawat, Sascha Schumann, Chris Scollo, Deepak Veliath. *PHP Programujeme profesionálně, 2. vydání*, Praha: Computer Press, 2002. ISBN 80-7226-310-2

[2] Luke Welling, Laura Thompsonová. *PHP a MySQL rozvoj webových aplikací, 2. vydání*, Praha: SoftPress, 2004. ISBN 80-86497-60-7

[3] Rob Allen, Nick Lo, Steven Brown. *Zend Framework in Action*, December 2008. ISBN: 1933988320

[4] RNDr. JUDr. Vladimír Šmíd, CSc. *Informační systémy,* `http://www.fi.muni.cz/~smid/mis-infsys.htm`

[5] Ing. Milan Šorm. *Webové informační systémy,* `http://is.mendelu.cz/dok_server/slozka.pl?id=14212&download=5241`, Brno: Ústav informatiky PEF MZLU, 2003.

[6] doc. Ing. Michal Krátký, Ph.D., Ing. Radim Bača, Ph.D. *Databázové systémy,* `http://db.cs.vsb.cz/edu/dbsys.pdf`, Ostrava: VŠB-Technická univerzita Ostrava, 2009.

[7] Zend Framework. *Oficiální webové stránky,* `http://framework.zend.com/`, Zend Technologies Ltd., 2009.

[8] Interval.cz. *Požadavky protokolu HTTP a jejich zpracování v PHP,* `http://interval.cz/`

[9] Wikipedia. *Content Management System,* `http://en.wikipedia.org/wiki/Content_management_system`

[10] Wikipedia. *Wiki,* `http://en.wikipedia.org/wiki/Wiki`

[11] Wikipedia. *Blog,* `http://cs.wikipedia.org/wiki/Blog`

[12] Wikipedia. *UML,* `http://cs.wikipedia.org/wiki/UML`

[13] Zend Framework. *Oficiální stránka projektu Zend Framework,* `http://framework.zend.com/`

[14] PostgreSQL. *Oficiální stránka databázového systému PostgreSQL,* `http://postgresql.org/`

[15] MySQL. *Oficiální stránka databázového systému MySQL,* `http://mysql.com/`

[16] freeCSStemplates.org. *Projekt zabývající se tvorbou CSS šablon,* `http://www.freecsstemplates.org/`

[17] CSS Zen Garden. *Projekt zabývající se tvorbou webového designu založeného na CSS,* `http://www.csszengarden.com/`

A Analýza – datový slovník

Atribut	Datový typ	Velikost	Klíč	Nulový	Index	IO	Popis
Tabulka: **users**							
idu	serial		Y	Y	Y		primární atribut
name	varchar	50	N	N	N		jméno
surname	varchar	50	N	N	N		příjmení
username	varchar	50	N	Y	N		přihlašovací jméno
email	varchar	50	N	N	N		kontaktní email
password	varchar	50	N	N	N		uživatelské heslo
created	timestamp		N	N	N		datum a čas vytvoření
updated	timestamp		N	N	N		datum a čas změny v záznamu
active	boolean		N	N	N		zablokování účtu uživatele
comment	varchar		N	N	N		komentář

Atribut	Datový typ	Velikost	Klíč	Nulový	Index	IO	Popis
Tabulka: **article_category_tree**							
idc	serial		Y	N	Y		primární klíč, jedinečné ID kategorie
top	integer		N	N	Y		úroveň kategorie
order	integer		N	N	N		pořadí kategorií, lze měnit
tree	string		N	N	Y		vygenerovaný strom pro stromovou strukturu

Atribut	Datový typ	Velikost	Klíč	Nulový	Index	IO	Popis
Tabulka: **article_category_name**							
idn	serial		N	N	N		primární klíč (uměle vytvořený)
idc	integer		N	N	N		cizí klíč, ID kategorie
idl	char	2	N	N	N		cizí klíč, ID jazykové mutace
idu	integer		N	N	N		cizí klíč, ID autora
name	varchar	1024	N	N	Y		jméno kategorie
comment	text		N	Y	N		komentář

Atribut	Datový typ	Velikost	Klíč	Nulový	Index	IO	Popis
Tabulka: **articles**							
ida	serial		Y	N	Y		primární klíč, ID článku
idc	integer		N	N	N		cizí klíč, ID kategorie
idl	char	2	N	N	N		cizí klíč, ID jazykové mutace
idu	integer		N	N	N		cizí klíč, ID autora
title	varchar		N	N	Y		titulek
text	text		N	N	Y		obsah článku
keywords	varchar		N	Y	Y		klíčová slova
description	varchar		N	Y	Y		popis

(*) – YYYY-MM-DD HH:mm

Atribut	Datový typ	Velikost	Klíč	Nulový	Index	IO	Popis
Tabulka: **languages**							
idl	varchar	2	Y	N	N	(*)	kód jazyka
lang	varchar	20	N	N	N		název jazyka

(*) – kódy zemí (např.: cs, en, de, it, ru, . . .)

Atribut	Datový typ	Velikost	Klíč	Nulový	Index	IO	Popis
Tabulka: **users_logging**							
idg	serial		Y	N	N		primární atribut, ID logu
idu	integer		N	N	N		cizí klíč, ID uživatele
date_login	timestamp		N	N	N	(*)	čas přihlášení
date_logout	timestamp		N	Y	N	(*)	čas odhlášení
ip	inet		N	Y	N	(**)	IP

(*) – YYYY-MM-DD HH:mm:ss
(**) – NNN.NNN.NNN.NNN/NN

B Obsah přiloženého CD

docs – uživatelský manuál, programátorská příručka

src – zdrojové soubory systému MicroCMS

sql – instalační skripty pro vytvoření databázových tabulek

these – text bakalářské práce